Linked Lists Implementation II

27 min

So far we can:

- create a new `LinkedList` using `.__init__()`
- get the head node of the list using `.get_head_node()`

Next up, we'll define methods for our `LinkedList` class that allow us to:

- insert a new head node
- return all the nodes in the list as a string so we can print them out in the terminal!

Instructions

1.

Define an `.insert_beginning()` method which takes `new_value` as an argument.

- Inside the method, instantiate a `Node` with `new_value`. Name this `new_node`.
- Now, link `new_node` to the existing `head_node`.
- Finally, replace the current `head_node` with `new_node`.

Note: Because the workspace is set up with spaces instead of tabs, you will need to use spaces to prevent Python from throwing an error. You can learn more about this here.
Hint
Remember that `Node` has a `.set_next_node()` class method you can use to set `new_node`'s `next_node`.

In the end, the method should look something like:

```
  def insert_beginning(self, new_value):
    new_node = Node(new_value)
    new_node.set_next_node(self.head_node)
    self.head_node = new_node
```

2.

Define a `.stringify_list()` method we can use to print out a string representation of a list's nodes' values.

The method should traverse the list, beginning at the head node, and collect each node’s value in a string. Once the end of the list has been reached, the method should return the string.

You can use `str()` to convert integers to strings!

Be sure to add `"\n"` between values so that each value prints on a new line.
Hint
Inside your `.stringify_list()` method body, start with a variable set to an empty string to collect all of the node values:

```
string_list = ""
```

You can use a `while` loop and `Node`’s `.next_node()` method to traverse the list.

If you accidentally get stuck in an infinite `while` loop, just use your browser to reload the page before editing your code (don’t worry — we’ve all been there).

3.

Test your code by uncommenting the print statement at the bottom of **script.py** — did your list print what you expected?

script.py

```
# We'll be using our Node class
class Node:
  def __init__(self, value, next_node=None):
    self.value = value
    self.next_node = next_node

  def get_value(self):
    return self.value

  def get_next_node(self):
    return self.next_node

  def set_next_node(self, next_node):
    self.next_node = next_node

# Our LinkedList class
class LinkedList:
  def __init__(self, value=None):
```

```python
    self.head_node = Node(value)

  def get_head_node(self):
    return self.head_node

# Add your insert_beginning and stringify_list methods below:

  def insert_beginning(self, new_value):
    new_node = Node(new_value)
    new_node.set_next_node(self.head_node)
    self.head_node = new_node

  def stringify_list(self):
    string_list = ""
    current_node = self.get_head_node()
    while current_node:
      if current_node.get_value() != None:
        string_list += str(current_node.get_value()) + "\n"
      current_node = current_node.get_next_node()
    return string_list


# Test your code by uncommenting the statements below - did your list print to the terminal?
ll = LinkedList(5)
ll.insert_beginning(70)
ll.insert_beginning(5675)
ll.insert_beginning(90)
print(ll.stringify_list())
```